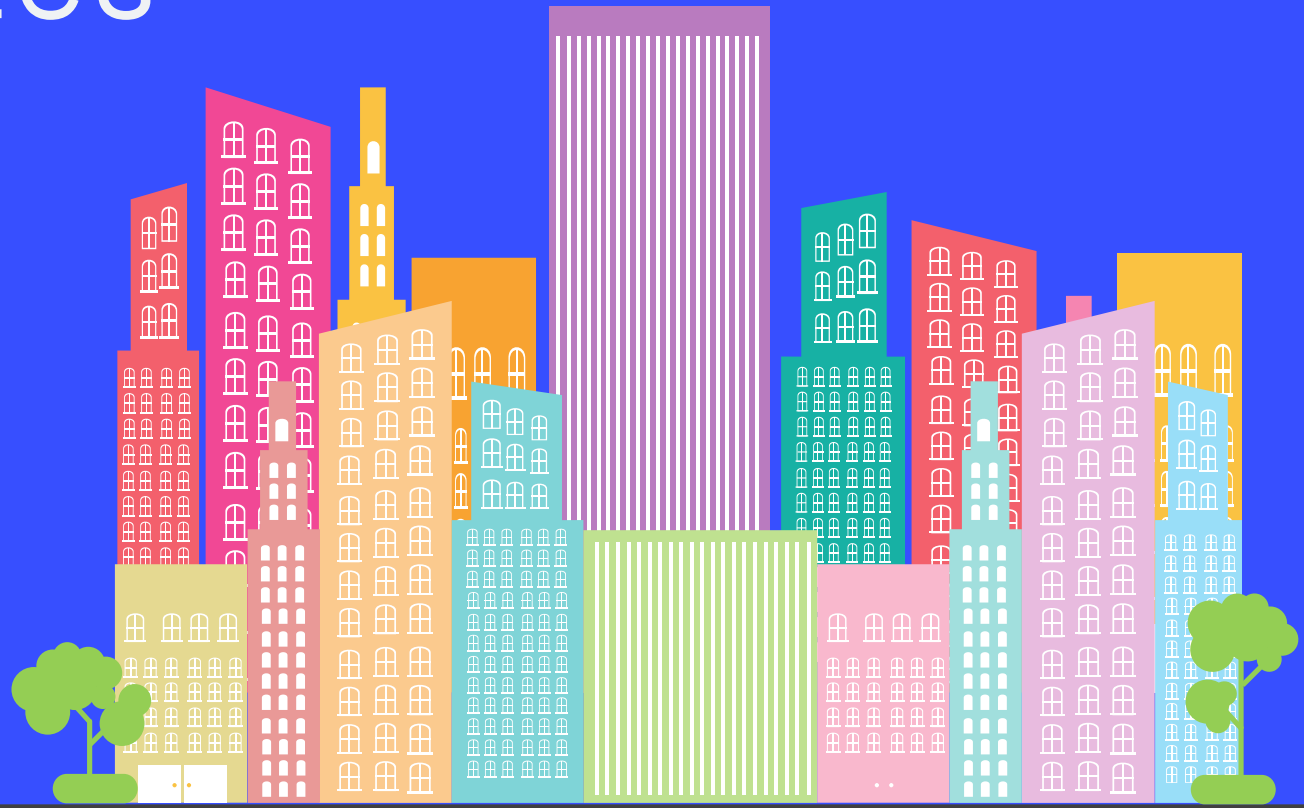
JS BUILDING BLOCKS

Values & Variables



GOALS

GOALS

GOALS

- Work with primitive types
- Understand let & const
- Use String Template Literals
- Work with common operators/methods

PRIMITIVE TYPES

- Number
- String
- Boolean
- Null
- Undefined

* Technically there are two others: Symbol and BigInt

Write Your Review

Click on a star to change your rating 1 - 5, where 5 = great! and 1 = really bad

Your Review:

Your Reivew

999 Characters remaining

Your Info:

Name:

Name

Email:

Email

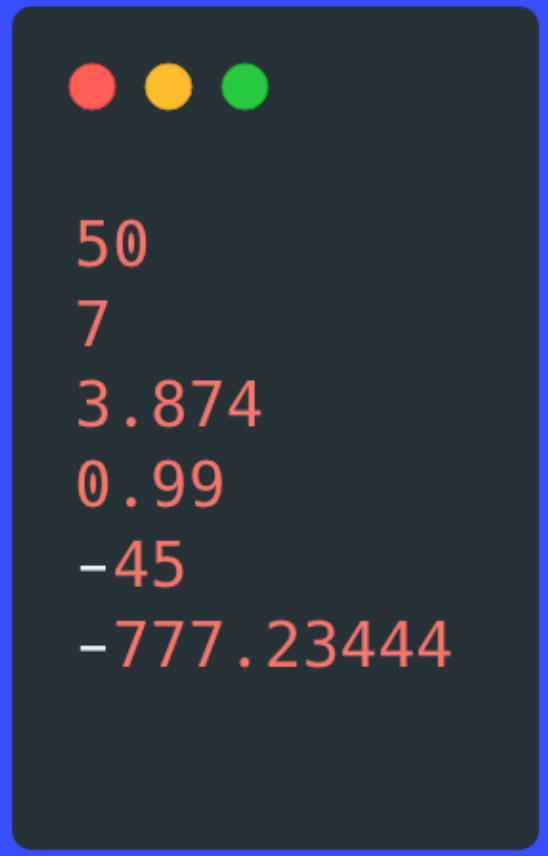
I Agree to the Terms blah blah blah

Submit

RUNNING CODE IN THE CONSOLE

We'll start by using the chrome console to quickly run code without any setup. (just for this section)

NUMBERS IN JS



- JavaScript has ONE Number type
- Positive numbers!
- Negative numbers!
- Whole number (integers)!
- Decimal numbers!

SIMPLE OPERATIONS

```
//Addition
50 + 5 //55

//Subtraction
90 - 1 //89

//Multiplication
11111 * 7 //77777

//Division
400 / 25 //16

//Modulo!!
27 % 2 //1
```

We have all the basic math operations you would expect...

// creates a comment, which JS will ignore

NaN

NOT A NUMBER

NaN is a numeric value that represents something that is not...a number

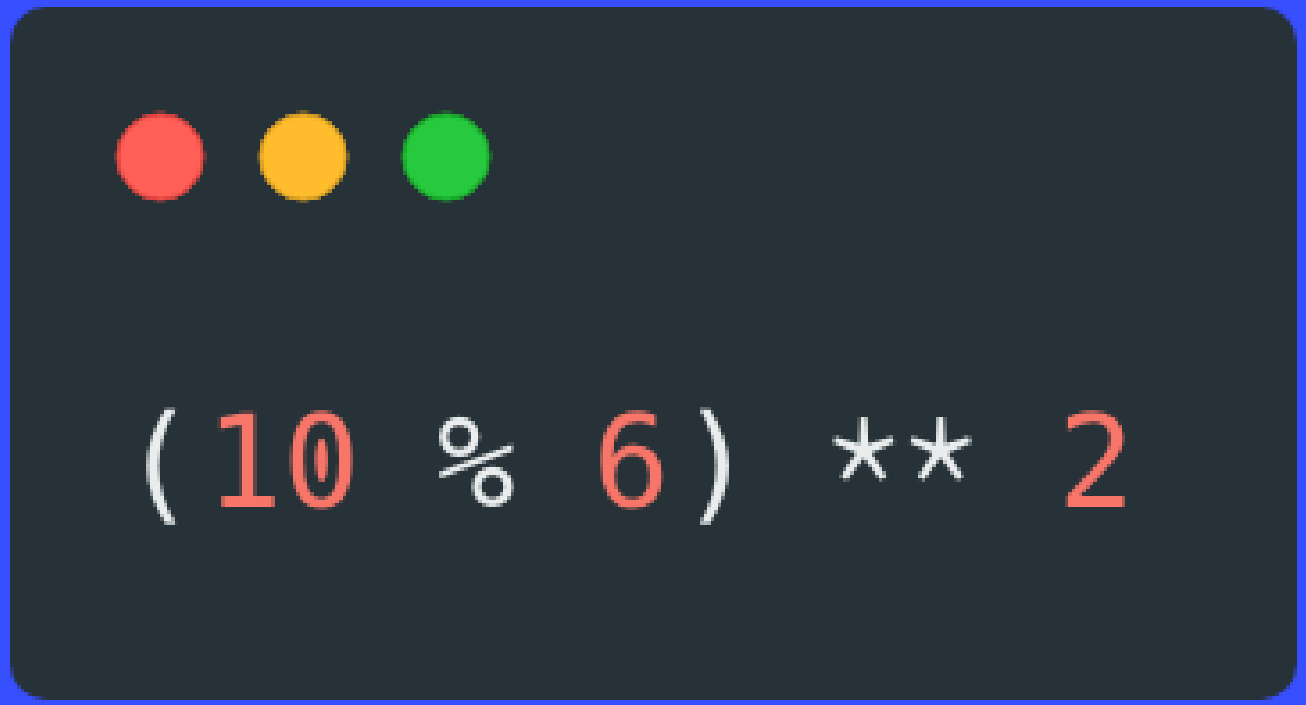
NaN

```
0/0 //NaN

1 + NaN //NaN
```

QUIZ

WHAT DOES THIS EVALUATE TO??

WHAT DOES THIS EVALUATE TO??



WHAT DOES THIS EVALUATE TO??

VARIABLES

Variables are like "labeled jars" for a value in JavaScript.

We can store a value and give it a name, so that we can...

- recall it
- use it
- or change it later on.

BASIC SYNTAX

```
let someName = value;
```

BASIC SYNTAX

```
let age = 55;
```

Make me a variable called "age" and give it the value of 55

RECALL VARIABLES

```
let hens = 4;

let roosters = 2;

hens + roosters //6
```

UPDATE VALUES

```
let hens = 4;

//A raccoon killed a hen :(
hens - 1; //3

hens; //Still 4!

//To actually change hens:
hens = hens - 1;
hens //3
```

This does not change the value stored in hens

This does!

CONST

```
const hens = 4;
hens = 20;  //ERROR!


const age = 17;
age = age + 1;  //ERROR!
```

const works just like let, except you CANNOT change the value

NOT ALLOWED!
YOU'RE IN TROUBLE!!
I'M TELLING MOM!!!

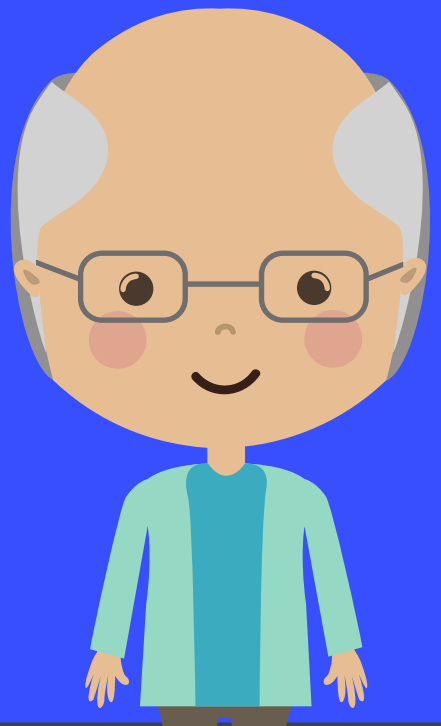
WHY USE CONST?

```
const pi = 3.14159;

const daysInWeek = 7;

const minHeightForRide  = 60;
```

Once we cover Arrays & Objects, we'll see other situations where *const* makes sense over *let*.

VAR

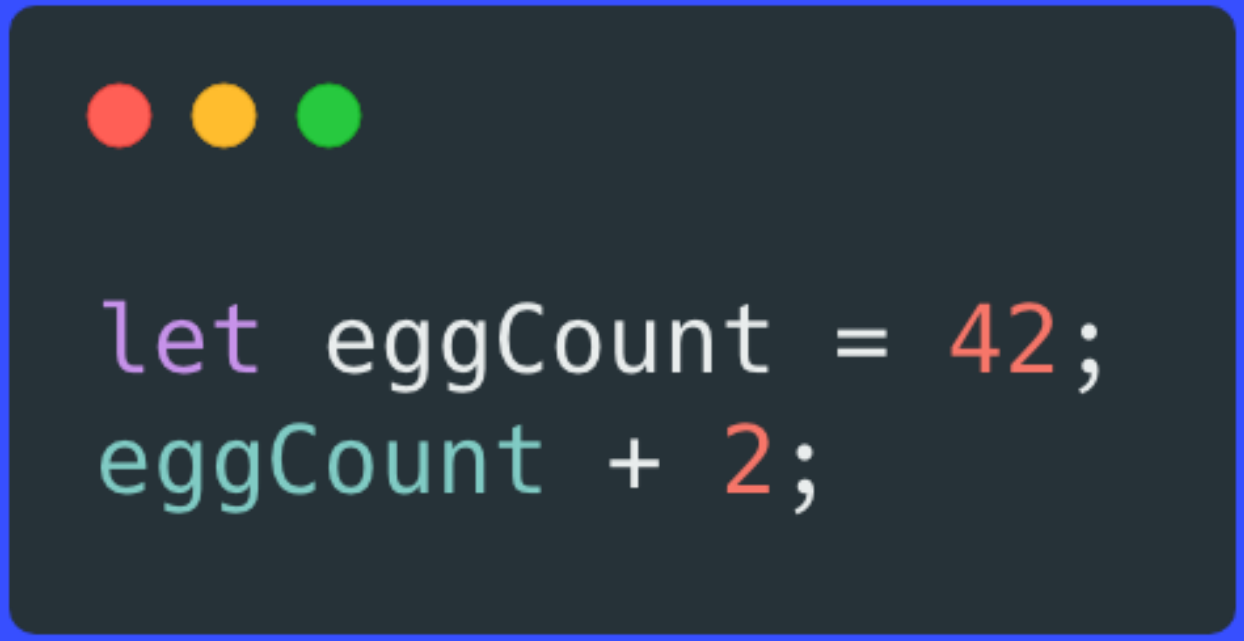
```
var tripDistance = 7.4;
```



Before *let* & *const*, *var* was the only way of declaring variables. These days, there isn't really a reason to use it.

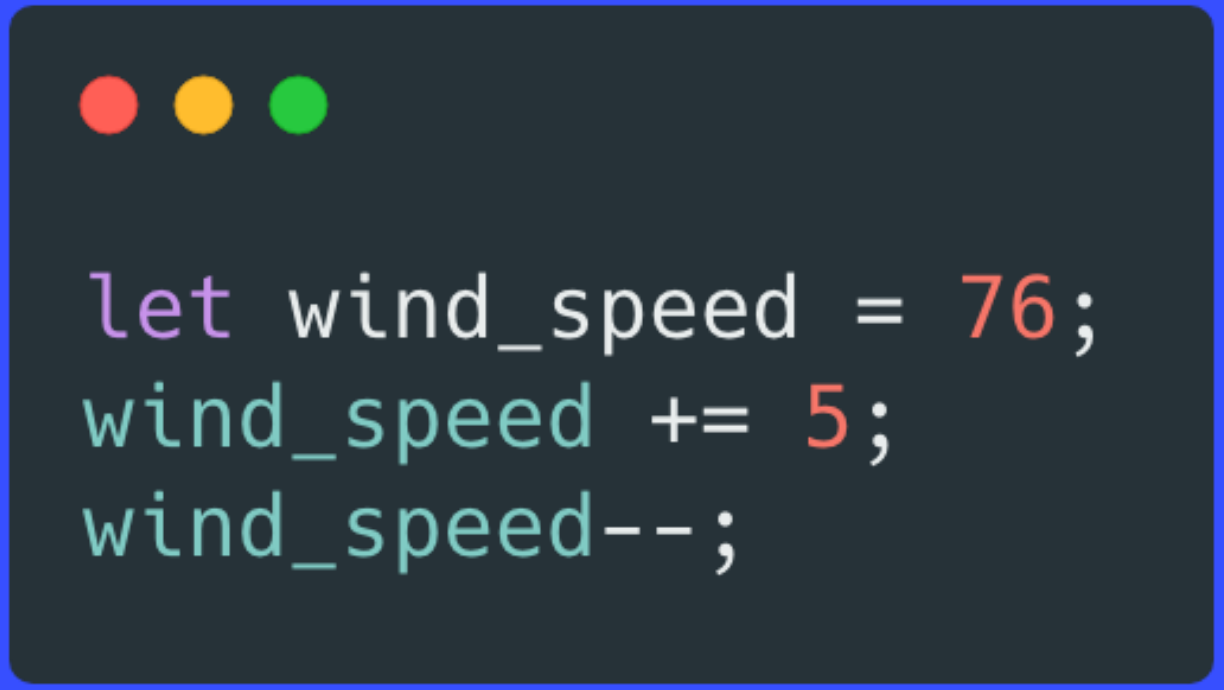
QUIZ

What is the value of *eggCount* ?



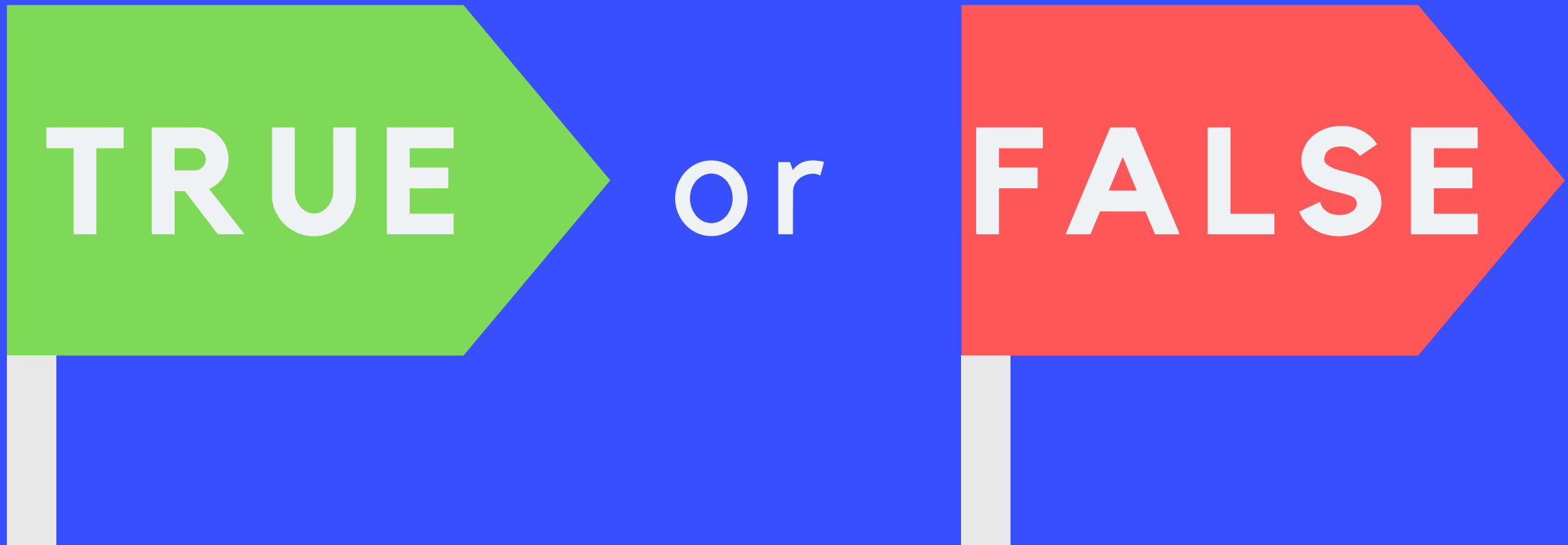
What is the value of *rating* after this code runs?

What's the value of *wind_speed* ?



What's the minor issue with this code?

BOOLEANS



BOOLEANS

```
let isLoggedIn = true;

let gameOver = false;

const isWaterWet = true;
```

Booleans are simple True or False values

Yes or No. 1 or 0.

VARIABLES CAN CHANGE TYPE

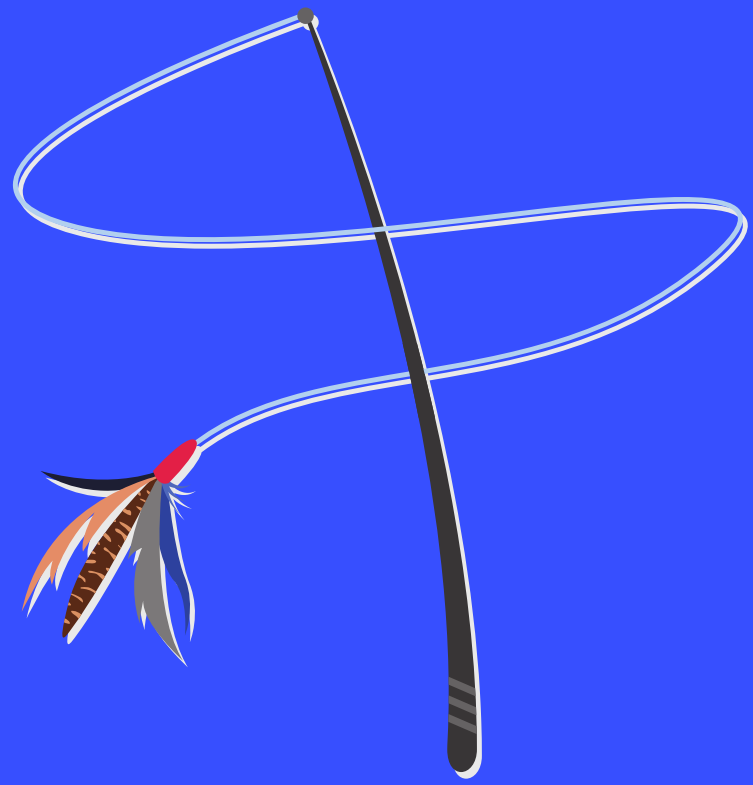
```
let numDonuts = 12; //it's a Number

numDonuts = false; //now it's a Boolean!

numDonuts = 129873872; //back to Number :)
```

You probably wouldn't change a number to a boolean, but you can!

STRINGS

In JavaScript,
Strings are pieces of text,
or *strings* of characters.

We wrap them in quotes



STRINGS

```
let firstName = "Ziggy";          Double quotes work

let msg = "Please do not feed the chimps!";

let animal = 'Dumbo Octopus';     So do single quotes

let bad = "this is wrong';     This DOES NOT work
```

Whether you use single or double quotes, just make sure you are consistent.

STRINGS ARE INDEXED

C	H	I	C	K	E	N
0	1	2	3	4	5	6

Each character has a corresponding index
(a positional number)

STRINGS ARE INDEXED

```
let firstName = 'Ziggy';

//Strings have a length property:
firstName.length //5

//Access individual characters using index:
firstName[0] // "Z"
firstName[3] // "g"

//Even though length is 5...
firstName[5] //DOES NOT WORK!
```

STRING METHODS

Strings come with a set of built-in methods, which are **actions** that can be performed on or with that particular string.

We can do things like...

- Searching within a string
- Replacing parts of a string
- Changing case (upper/lowercase)

thing.method()

Changing Case

```
let msg = 'I am king';
let yellMsg = msg.toUpperCase(); // 'I AM KING'

let angry = 'LeAvE mE aLoNe!';
angry.toLowerCase(); // 'leave me alone!'

//the value in angry is unchanged
angry; // 'LeAvE mE aLoNe!'
```

trim

```
let greeting = '    leave me alone plz     ';

greeting.trim() // 'leave me alone plz'
```

thing.method(arg)

Some methods accept **arguments** that modify their behavior.

We pass these arguments inside of the parentheses.

indexOf

```
let tvShow = 'catdog';

tvShow.indexOf('cat'); // 0
tvShow.indexOf('dog'); // 3
tvShow.indexOf('z'); // -1 (not found)
```

slice

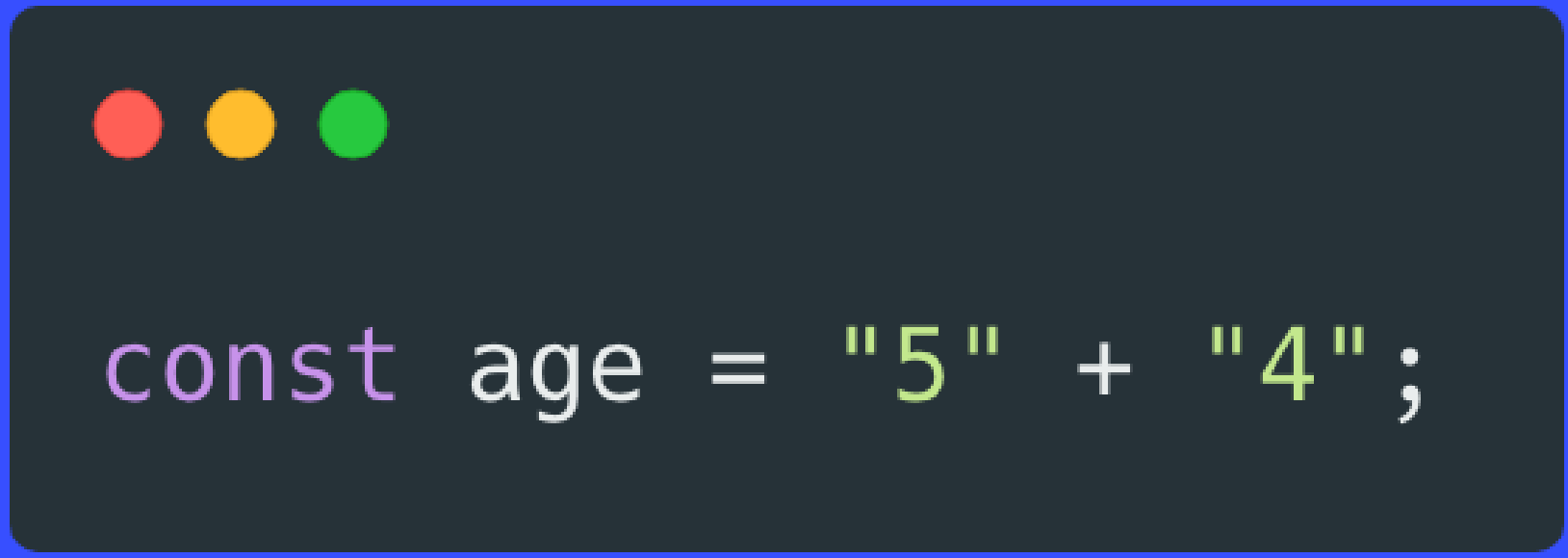
```
let str = 'supercalifragilisticexpialidocious'

str.slice(0,5); //'super'

str.slice(5); // 'califragilisticexpialidocious'
```

replace

```
let annoyingLaugh = 'teehee so funny! teehee!';

annoyingLaugh.replace('teehee', 'haha') // 'haha so funny! teehee!'
//Notice that it only replaces the first instance
```

QUIZ

WHAT IS THE VALUE OF *AGE*?



WHAT DOES THIS EVALUATE TO?

WHAT DOES THIS EVALUATE TO?

What is the value of *song*?

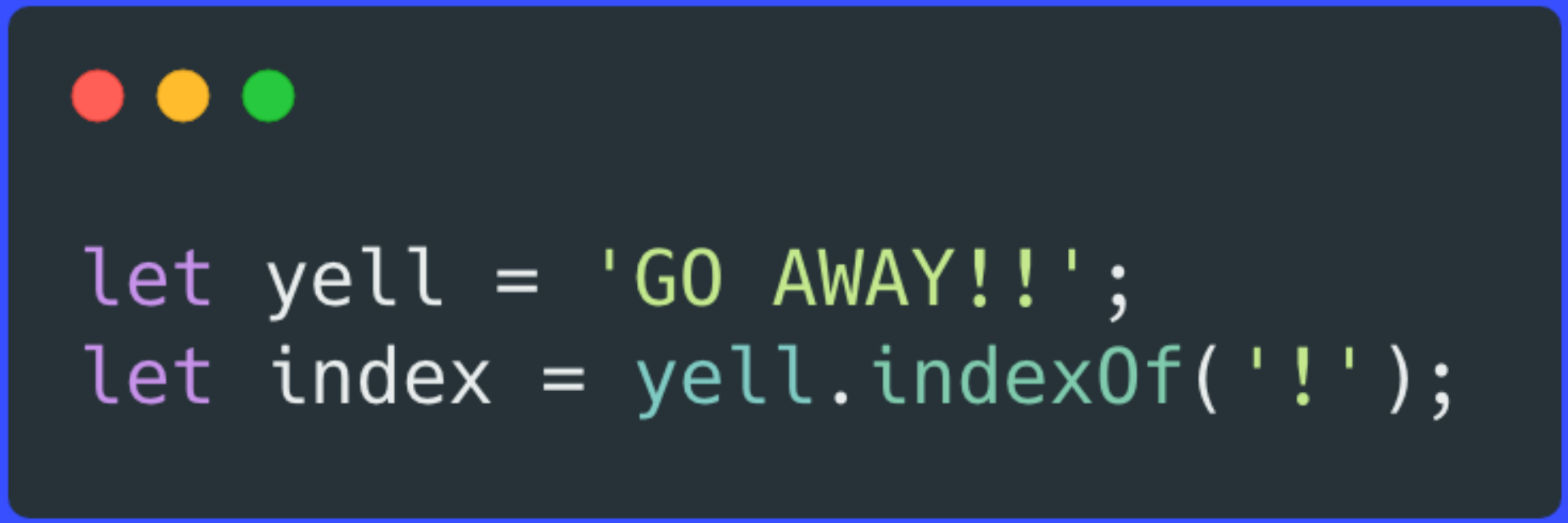
```
let song = "london calling";
song.toUpperCase();
```

What is the value of *cleanedInput*?

```
let userInput = "   TODD@gmail.com";
let cleanedInput = userInput.trim().toLowerCase();
```

What is the value of *index*?

What is the value of *index*?



WHAT DOES THIS EVALUATE TO?

```
'GARBAGE!'.slice(2).replace("B",'');
```

STRING ESCAPES

- \n – newline
- \' – single quote
- \" – double quote
- \\ – backslash

STRING TEMPLATE LITERALS

Template literals are strings that allow embedded expressions, which will be evaluated and then turned into a resulting string.

```
`I counted ${3 + 4} sheep`;  // "I counted 7 sheep"
```

WE USE BACK-TICKS NOT SINGLE QUOTES

`I am a template literal`

* The back-tick key is usually above the tab key

TEMPLATE LITERALS

```
let username = 'Ziggy31';
`Welcome back, ${username}` // "Welcome back, Ziggy31"

`GAME OVER ${username.toUpperCase()}` // "GAME OVER ZIGGY31"
```

TEMPLATE LITERALS

```
let item = 'cucumbers';
let price = 1.99;
let quantity = 4;

`You bought ${quantity} ${item}, total price: $${price*quantity}`;
//"You bought 4 cucumbers, total price: $7.96"
```

NULL & UNDEFINED

- Null
  - "Intentional absence of any value"
  - Must be assigned
- Undefined
  - Variables that do not have an assigned value are undefined

NULL

```
1 // No one is logged in yet...
2 let loggedInUser = null;  //value is explicitly nothing
3
4 // A user logs in...
5 loggedInUser = 'Alan Rickman';
```

UNDEFINED

```
1 let pickles;  //We didn't assign a value
2 pickles; //undefined,
3 pickles = 'are very gross'
4
5 //Undefined also comes up in other situations:
6 let food = 'tacos';
7 food[7]; //undefined
```